Inferring connectivity from imaging activity

Joshua Vogelstein, some others, Liam Paninski

April 29, 2009

Abstract

Simultaneously imaging small populations of neurons using calcium sensors is now becoming routine in labs across the world. We have developed analytical tools to maximally utilize these beautiful data sets. In particular, the following neurobiological questions are of interest: (i) what are the response properties of populations of neurons that are in close physical proximity, (ii) what is the functional circuit underlying these response properties. Previously, we built a forward model characterizing the relationship between stimuli, spike trains, intracellular calcium concentration, and fluorescence time series observations; and then inverted that model using non-linear state-space methods (i.e., a particle-filter-smoother adapted for this model, embedded in an expectation-maximization algorithm).

Here, that forward model is generalized to allow for dependencies between neurons (i.e., cross-coupling terms), and the inference and learning algorithm is appropriately modified as well. We show that given only a short movie (¡ 10 min), and reasonable assumptions on noisiness, spike rates, and coupling strengths, we can accurately reconstruct circuits governing small populations (e.g., 10 neurons; see S1). As the number of neurons increases, or the data quality decreases, we can impose experimentally justifiable priors on the distribution of coupling weights, to improve our estimates.

We are currently pursuing confirming our parameter estimates using in vitro preparations. In particular, by simultaneously imaging a population of neurons, and recording electrophysiologically from at least one, we can validate our inferred connection strengths. Our hope is that they algorithms developed here will be useful in a wide array of experiments and preparations, especially in vivo calcium imaging.

1 Introduction

Our goal is to estimate the connection matrix of a population of neurons from simultaneously imaging their activity.

2 Model

2.1 Single neuron model

We assume that we have a discrete time model, with time step size Δ and T total time steps. In each time step, the neuron can emit any non-negative number of spikes, i.e., $n_t \in \{0, 1, 2, \cdots\} = \mathbb{N}_0$. The spiking is governed by a Poisson process with rate $\lambda_t \Delta$, where λ_t is given by:

$$\lambda_t = \exp\{b + \boldsymbol{k}'\boldsymbol{s}_t + \omega h_t\} \tag{1}$$

$$h_t - h_{t-1} = -\frac{\Delta}{\tau_h} h_t + n_{t-1} + \sigma_h \sqrt{\Delta}\varepsilon \tag{2}$$

calcium model:

$$[\text{Ca}^{2+}]_t - [\text{Ca}^{2+}]_{t-1} = -\frac{\Delta}{\tau_c}([\text{Ca}^{2+}]_t + [\text{Ca}^{2+}]_b) + A n_t + \sigma_c \sqrt{\Delta}\varepsilon, \tag{3}$$

observation model:

$$F_t = \alpha S([\text{Ca}^{2+}]_t) + \beta + (S([\text{Ca}^{2+}]_t) + \sigma_F)\varepsilon_t \tag{4}$$

where $S(x) = x^n/(x^n + k_d)$ is the standard Hill equation, the ε's indicate standard normal random variables.

2.2 Multiple neuron model

We assume the firing rate for neuron i, $\lambda_{i,t}\Delta$, is given by:

$$\lambda_{i,t} = f(b_i + \boldsymbol{k}_i'\boldsymbol{s}_t + \sum_{j=1}^{N}\omega_{ij}h_{j,t}) \tag{5}$$

$$h_{i,t} - h_{i,t-1} = -\frac{\Delta}{\tau_{h_i}}h_{i,t} + n_{i,t-1} + \sigma_{h_i}\sqrt{\Delta}\varepsilon, \tag{6}$$

where each h_i corresponds to the spike history term associated with neuron i, and has an associated time-constant, τ_i. The input each neuron recieves, therefore, is the sum of spike history terms from all neurons (including itself). Thus, $\boldsymbol{\omega} = \{\omega_{ij}\} = \{\omega_{11}, \omega_{12}, \ldots, \omega_{1N}, \ldots, \omega_{NN}\}$ comprises the weight matrix for these neurons.

calcium model:

$$[\text{Ca}^{2+}]_{i,t} - [\text{Ca}^{2+}]_{i,t-1} = -\frac{\Delta}{\tau_{c_i}}([\text{Ca}^{2+}]_{i,t} + [\text{Ca}^{2+}]_{b_i}) + A_i n_{i,t} + \sigma_{c_i}\sqrt{\Delta}\varepsilon, \tag{7}$$

observation model:

$$F_{i,t} = \alpha_i S([\text{Ca}^{2+}]_{i,t}) + \beta_i + (S([\text{Ca}^{2+}]_{i,t}) + \sigma_{F_i})\varepsilon, \tag{8}$$

Note that we assume that $S(\cdot)$ is the same for each neuron (i.e., the nonlinear function is the same for each neuron, because the parameters are a function of the indicator, not the neuron). Further note that we assume (for now) that the noise on the calcium and observation for each neuron is independent (an assumption that we will probably want to relax later).

3 Mathematical Background

To estimate $\boldsymbol{\omega}$, we adopt a state-space framework. More specifically, let y_t be the observations, and x_t be the *hidden states*. We then have:

$$\boldsymbol{y}_t \sim g_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t) \tag{9}$$

$$\boldsymbol{x}_t \sim f_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{10}$$

where $g_{\boldsymbol{\theta}}(\cdot)$ and $f_{\boldsymbol{\theta}}(\cdot)$ indicate the *observation* and *transition* distributions, respectively (and we have adopted the shorthand notation $f_{\boldsymbol{\theta}}(\cdot) = f(\cdot|\boldsymbol{\theta})$. Letting $\boldsymbol{X}_t = \{X_{i,t}\} = \{X_{1,t}, \ldots, X_{N,t}\}$, for the above model, we have $\boldsymbol{y}_t = \boldsymbol{F}_t$, $\boldsymbol{x}_t = \{\boldsymbol{n}_t, \boldsymbol{h}_t, [\textbf{Ca}^{2+}]_t\}$, and $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\tau_c}, [\text{Ca}^{\boldsymbol{2+}}]_{\boldsymbol{b}}, \boldsymbol{\sigma_c}, \boldsymbol{b}, \{\boldsymbol{k}_i\}, \boldsymbol{\omega}, \boldsymbol{\tau_h}, \boldsymbol{\sigma}_h\}$. Note that given the above model, $\boldsymbol{y}_t \in \Re_+^N$, i.e., each y_t is a non-negative real number, and $\boldsymbol{x}_t \in \Re_+^{2N} \times \{0,1\}^N$, i.e., $\boldsymbol{x}_t$ lies in a $3N$-dimensional space, where $2N$ dimensions are non-negative real numbers ($[\text{Ca}^{2+}]_t$ and h_t for each neuron), and the rest are binary ($n_t \in \{0,1\}$ for each neuron). Thus, we have a "hybrid" system, in that some of our hidden states are continuous, and others are discrete. We may now formally state our goal as:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmax}}\, P_{\boldsymbol{\theta}}(\boldsymbol{y}) = \underset{\boldsymbol{\theta}}{\text{argmax}} \int P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x})d\boldsymbol{x} \tag{11}$$

where, we have introduced the notation, $\boldsymbol{y} = \boldsymbol{y}_{0:T}$, and $P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x})$ may be factored as:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) = \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_0)\prod_{t=1}^{T} g_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t)f_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{12}$$

where $\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_0)$ is the initial distribution of $\boldsymbol{x}$. Sadly, for our model, integral in (11) is computationally intractable. Therefore, we adopt an EM approach:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}') \tag{13}$$

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}') = E_{p_{\boldsymbol{\theta}'}(\boldsymbol{x}|\boldsymbol{y})} \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) = \int p_{\boldsymbol{\theta}'}(\boldsymbol{x}|\boldsymbol{y}) \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) d\boldsymbol{x} \tag{14}$$

Because we have a state-space model, the above integral may be simplified:

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}') = \int p_{\boldsymbol{\theta}'}(\boldsymbol{x}_0|\boldsymbol{y}) \times \ln \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_0) d\boldsymbol{x}_0 + \\ \sum_{t=1}^{T} \iint p_{\boldsymbol{\theta}'}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}|\boldsymbol{y}) \times \ln f_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) d\boldsymbol{x}_t d\boldsymbol{x}_{t-1} + \\ \sum_{t=0}^{T} \int p_{\boldsymbol{\theta}'}(\boldsymbol{x}_t|\boldsymbol{y}) \times \ln g_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t) d\boldsymbol{x}_t \end{aligned} \tag{15}$$

As we have a nonlinear observation model, and nonlinear transition model, the integrals in (15) are intractable as well, so they must be approximated. We adopt a monte carlo strategy, in which we approximate the above integrals with sums. This requires approximating the continuous valued hidden states with discrete valued hidden states. The key is to be able to estimate the pairwise joint conditionals:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}|\boldsymbol{y}) \approx \sum_{k,m=1}^{k} \widetilde{p}_{\boldsymbol{\theta}}\big(\boldsymbol{x}_t^{(k)}, \boldsymbol{x}_{t-1}^{(m)}|\boldsymbol{y}\big) \delta_{\boldsymbol{x}_t^{(k)}}(\boldsymbol{x}_t) \delta_{\boldsymbol{x}_{t-1}^{(m)}}(\boldsymbol{x}_{t-1}) \tag{16}$$

where $\delta_{\boldsymbol{x}}(\cdot)$ denotes the Dirac mass at point $\boldsymbol{x}$, and $\widetilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(k)}, \boldsymbol{x}_{t-1}^{(m)}|\boldsymbol{y})$ is computed pointwise for each $\boldsymbol{x}_t^{(k)}$ and $\boldsymbol{x}_{t-1}^{(m)}$, and then normalized such that $\sum_{k,m=1}^{k} \widetilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(l)}|\boldsymbol{x}_{t-1}^{(m)}) = 1$. From this approximation, we can also estimate the marginal conditionals and initial distribution, by integrating out $\boldsymbol{x}_{t-1}$. The key is to efficiency discretize the space, so L is small and the approximation is still good. We adopt a Monte Carlo strategy, in which we represent the pairwise joint conditionals as a product of terms:

$$\widehat{P}_{\boldsymbol{\theta}}(F_v|[\text{Ca}^{2+}]_t)\big(\boldsymbol{x}_t^{(l)}, \boldsymbol{x}_{t-1}^{(m)}|\boldsymbol{y}\big) = \widehat{P}_{\boldsymbol{\theta}}(F_v|[\text{Ca}^{2+}]_t)\big(\boldsymbol{x}_t^{(l)}|\boldsymbol{y}\big) \frac{\widetilde{f}_{\boldsymbol{\theta}}\big(\boldsymbol{x}_t^{(l)}|\boldsymbol{x}_{t-1}^{(m)}\big) w_{t-1}^{(m)}}{\sum_{m=1}^{L} \widetilde{f}_{\boldsymbol{\theta}}\big(\boldsymbol{x}_t^{(l)}|\boldsymbol{x}_{t-1}^{(m)}\big) w_{t-1}^{(m)}} \tag{17}$$

where $w_t^{(l)}$ is defined by

$$P_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{l=1}^{L} w_t^{(l)} \delta_{\boldsymbol{x}_{0:t}^{(l)}}(\boldsymbol{x}_{0:t}) \tag{18}$$

The maximize efficiency, therefore, we aim to choose $\boldsymbol{x}_t^{(l)}$'s to make their associated weights, $w_t^{(l)}$'s approximately equal. In the next section, we describe a number of algorithms, with increasing complexity and (hopefully) utility.

Thus, it should be clear from the above that ideally, one would ideally like to sample jointly from $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}_t)$. As this is so high dimensional (more specifically, $3N$ dimensions), the importance sampling idea breaks down (as it is incapable of approximating a high dimensional space with a small number of particles). Therefore, we develop a number of approaches to combat this curse of dimensionality.

3.1 population dynamics

plugging model described in section 2.2 into (9) and (10), we have:

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \prod_{i=1}^{N} p([\text{Ca}^{2+}]_{i,t}|[\text{Ca}^{2+}]_{i,t-1}, n_{i,t}) p(n_{i,t}|\boldsymbol{h}_t) p(h_{i,t}|h_{i,t-1}, n_{i,t-1}) \tag{19}$$

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t) = \prod_{i=1}^{N} p(F_{i,t}|[\text{Ca}^{2+}]_{i,t}) \tag{20}$$

where $\boldsymbol{h}_t = h_{1,t}, \ldots, h_{N,t}$ and the above distributions are defined by

$$p([\text{Ca}^{2+}]_{i,t}|[\text{Ca}^{2+}]_{i,t-1}, n_t) = \mathcal{N}([\text{Ca}^{2+}]_{i,t}; a_{c_i}[\text{Ca}^{2+}]_{i,t-1} + b_i + A_i n_{i,t}; \sigma_{c_i}^2 \Delta) \tag{21}$$

$$p(n_{i,t}|\boldsymbol{h}_{i,t}) = \mathcal{B}(n_{i,t}; \lambda_{i,t}) \tag{22}$$

$$p(h_{i,t}|h_{i,t-1}, n_{i,t-1}) = \mathcal{N}(h_{i,t}; a_{h_i} h_{i,t-1} + n_{i,t-1}, \sigma_{h_i}^2 \Delta) \tag{23}$$

$$p(F_{i,t}|[\text{Ca}^{2+}]_{i,t}) = \mathcal{N}(F_{i,t}; \alpha_i S([\text{Ca}^{2+}]_{i,t}) + \beta_i, (S([\text{Ca}^{2+}]_{i,t}) + \sigma_{F_i})^2) \tag{24}$$

and we have introduced $a_{c_i} = 1 - \Delta/\tau_{c_i}$, $b_i = -\Delta[\text{Ca}^{2+}]_{b_i}/\tau_{c_i}$, and $a_{h_i} = 1 - \Delta/\tau_{h_i}$, for brevity.

4 Results

4.1 Main Result

Our goal is to infer the probability of each neuron spiking at any time, conditioned on *all* the fluorescence measurements from the entire population of neurons: $P_{\boldsymbol{\theta}}(\boldsymbol{n}_t|\boldsymbol{F}_{1:T})$ for $t = 1 \ldots, T$. As the uncertainty is non-Gaussian (due to the assumed Bernoulli distribution governing spikes), and the dynamics are nonlinear (due to the saturation of fluorescence, $S(\cdot)$, standard linear-Gaussian filtering is inadequate for this model. Instead, we generalize our previous result [**?**], in which we utilized sequential Monte Carlo methods to develop a model-based optimal nonlinear filter. Very briefly, in that work, we used a particle filter to recursively infer $P_{\boldsymbol{\theta}}(n_t|F_{1:t})$, for $t = 1, \ldots, T$, and used a particle smoother to recurse backwards, obtaining $P_{\boldsymbol{\theta}}(n_t|F_{1:T})$ [**?**]. By embedding this particle-filter-smoother (PFS) into an expectation-maximization framework, we can iterate inferring the hidden spike trains and learning the parameters [**?**]. Importantly, in that work, we modeled each neuron as an independent generalized linear model (GLM). In contrast, here we model the neurons as coupled GLMs [**?**, **?**, **?**]. Thus, we can condition the probability of each neuron spiking at any time on the spike history of all observed neurons. More precisely, we proceed as described in Algorithm 1. Note, we have assumed that all the parameters governing $\boldsymbol{F}$ and $\boldsymbol{C}$ have been estimated already, using the GLM PFS described in [**?**].

Algorithm 1 Pseudocode for inference and learning for a population of simultaneously observed neurons.

```
1:  while $j = 1, \ldots$ do {$j$ indexes iterations}
2:    for $i = 1$ to $m$ do {$i$ indexes neurons}
3:      Let $\widetilde{\boldsymbol{x}}_{i,t}^{(j)} = [\boldsymbol{x}_t, \widetilde{\boldsymbol{h}}_{\backslash i,t}^{(j)}]'$, where $\widetilde{\boldsymbol{h}}_{\backslash i,t}^{(j)} = E[\boldsymbol{h}_{\backslash i,t}^{(j)}|\boldsymbol{F}_{1:T}]$ and $\boldsymbol{h}_{\backslash i,t} = \{h_{1,t}, \ldots, h_{i-1,t}, h_{i+1,t}, \ldots, h_{m,t}\}$
4:      Let $\widetilde{\boldsymbol{k}}_i^{(j)} = [\boldsymbol{k}_i^{(j)}, \boldsymbol{\omega}_{\backslash \boldsymbol{i}}^{(j)}]'$, where $\boldsymbol{\omega}_{\backslash \boldsymbol{i}} = \{\omega_{i,1}, \ldots, \omega_{i,i-1}, \omega_{i,i+1}, \ldots, \omega_{i,m}\}$
5:      Infer $P_{\boldsymbol{\theta}^{(j)}}(\{C, n, h\}_{i,t}^{(j+1)}|\boldsymbol{F}_{1:T})\ \forall t$ where $n_{i,t} \sim \text{Ber}(n_{i,t}; (\widetilde{\boldsymbol{k}}_i^{(j)})'\widetilde{\boldsymbol{x}}_{i,t}^{(j)})$ using GLM PFS as described in [?]
6:    end for
7:    for $i = 1$ to $m$ do
8:      Estimate $\widetilde{\boldsymbol{k}}_i^{(j+1)} = [\boldsymbol{k}_i^{(j+1)}, \boldsymbol{\omega}_{\backslash \boldsymbol{i}}^{(j+1)}]$ using GLM PFS as described in [?]
9:    end for
10: end while
```

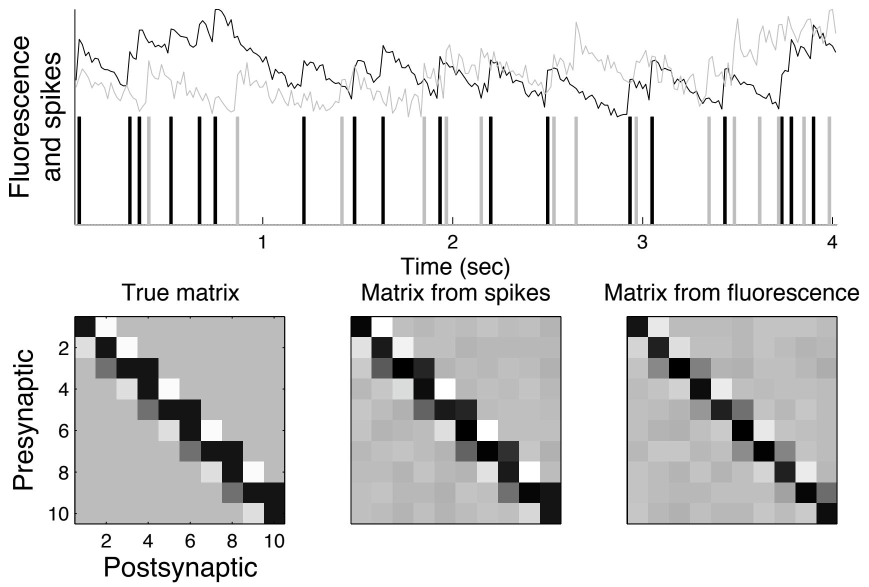

Figure 1: Inferring network connectivity given noisy simulated calcium fluorescence data. Our main result is that we can infer network connectivity given only short sequences of observations (< 10 min), for small populations of neurons (~ 10), given reasonable assumptions on spike rate (~ 5 Hz), observation frequency (60 fps), and observation noise. **Top**: True spikes and observed noisy calcium fluorescence traces for two cells in the simulated network. Calcium data sampled at 60 Hz. (Only four seconds of a longer experiment are shown here, for clarity.) **Bottom**: True connectivity matrix; connectivity matrix estimated given the fully observed spikes; and connectivity matrix estimated given just the noisy calcium data, given about 2000 spikes/neuron. White denotes excitatory connections; black denotes inhibition; the same grayscale map is used for each panel. A small simulated network was used for illustration purposes. Each cell inhibited itself following each spike (a relative refractory effect) and excited or inhibited its nearest neighbors; the strength and time constant of these excitatory terms were chosen to mimic the experimentally determined parameters (a spike in one cell modulates the instantaneous firing rate in another cell by no more than 5% – 10%). No external drive was applied to the network; i.e., only spontaneous network dynamics are observed here. Note that it is somewhat harder to estimate the connectivity map given noisy calcium data than the true spike trains (specifically, some of the inhibitory connections are slightly underestimated in the right panel); nonetheless the correct connectivity is obtained with only a few minutes of calcium data.

4.2 Embedded MCMC improves inference

To test the performance for inference of neural connectivity in a neural population, we simulated such inference in conditions close to such expected in real data. Specifically, we solve connectivity inference for a network of simulated spiking neurons, constructed closely following empirical data known about the real neural networks in the cortex. We assume a population of neurons that are spontaneously firing action potentials. While it is known, that the functional connectivity weights in general do not properly reflect anatomical connectivity in a circuit, we will show that for the above system of spontaneously firing neurons the direct correspondence, indeed, exists, and anatomical connectivity is properly recovered via functional connectivity weights.

Functional connectivity may fail to faithfully represent anatomical circuit structure if false correlations are present between different neurons, induced e.g. by common inputs, or if the dynamics of neural population is entirely concentrated on a low-dimensional subspace of the full configurational space H. Note that these two statements are, in a sense, different ways of stating the same condition: if activity of different neurons is tightly correlated, their dynamics is concentrated on a low-dimensional plane; and vice-versa - concentration of dynamics onto a low-dimensional plane will be perceived as correlation in activity of different neurons. (In turn, low dimensionality of the neural dynamics may be caused by different factors, including common input, small subset of command neurons driving the circuit, or even emergent property of a network.) Low dimensionality of the neural dynamics results in that the inference problem Eq.(**??**) becomes underdetermined, i.e. there may exist directions in W along which connectivity is not constrained by activity data (i.e. directions orthogonal to the subspace of all observed neural activity configurations), or is poorly constrained. This, naturally, leads to W being poorly defined along these directions. The necessary condition for good correspondence between functional connectivity weights W and anatomical connectivity, therefore, is *full-dimensionality* of the observed neural dynamics. In case of spontaneously firing system of neurons this condition is, in fact, satisfied by many independent neuron-ignitions, thus, fully sampling possible directions in the configurational space H. If spontaneously active preparation by itself fails to display sufficient degree of independence between randomly firing neurons (e.g. if low-dimensionality of the activity subspace is the emergent property of studied circuit), such pattern may be induced by randomly activating subsets of neurons via ChR2 [...] or glutamate uncaging.

We also note that the correlations induced by secondary and so on synaptic transmissions (such as when neuron A results in firing of neuron B, which in turn results in firing by neuron C), are all properly resolved in GLM-fitting process via the so called explaining-away process. In other words, because we do not just identify correlations between neural firings with the functional connectivity weights $w^{kk'}$, but instead statistically fit a model of neural interactions, if found weights between neurons A and B, and B and C are sufficient to explain the correlation between A and C, the weight connecting A and C will not appear in the model - the correlation between A and C was "explained away" by correlations between A and B, and B and C. By this, the multi-synaptic firing patterns do not confuse our estimation process.

We prepared a small network of $N = 50$ neurons randomly sparsely connected, and firing stochastically with the base firing rate of about 5Hz. Each neuron was modeled with GLM as a linear-nonlinear driven Poisson spike generator, as described above

$$\begin{aligned} P(n^k_{t+1}|\{h_{t+1}\}) =& \text{Poiss}(\lambda^k_{t+1}\Delta t) \\ \lambda^k_{t+1} = g(J^k_t) =& g(b_0 + \textstyle\sum_{\tau>0} w^{k,k'}_\tau n^{k'}_{t-\tau}), \end{aligned} \tag{25}$$

with exponential transfer function $g(J) = \exp(J)$. We assumed no external modulating input $X_{ext}(t)$.

The network was divided into excitatory and inhibitory components. Neurons in such components were either entirely excitatory or inhibitory; i.e., all connections outgoing from such neuron were either all simultaneously positive (excitatory) or negative (inhibitory). Excitatory neurons were randomly connected with each other and the inhibitory neurons with probability $f_c = 0.1$ (observed probability for a local connection between nearby neurons in the cortex [**]). The synaptic weight of each connection v, as defined by max EPSP amplitude, were generated from exponential distribution with mean $0.5\mu V$ [**] (we neglected here the "heavy tail" of the distribution of synaptic weights observed in some datasets [**]). While synaptic weights are typically measured in PSP units of μV, in GLM model connections are measured in log-rate units of Eq.(25). In other words, GLM weights describe the *change in probability of neuron k to fire given neuron k′ has firing before*. By utilizing this definition, we may convert a PSP weight v_{EPSP} for a neuron V_{base} below spike-triggering threshold into the GLM weight as $v_{GLM} \approx v_{EPSP}/V_{base}$. In other words, V_{base}/v_{EPSP} spikes are required to push neuron over the threshold. Given the definition of the firing rate in (25), this leads to the following equation for the log-rate couplings $w^{kk'}$

$$w^{kk'} = \ln(-\ln(\exp(-f^k\Delta t) - v^{kk'}_{EPSP}/V_{base})/\Delta t/f^k), \tag{26}$$

where f^k is the base "stochastic" firing rate of neuron k.

20% of all neurons were taken to be inhibitory [**]. Interneurons were randomly connected among themselves and to excitatory neurons with the same frequency $f_c = 0.1$ as above. The strength of the inhibitory connection was drawn from the exponential distribution with mean chosen to balance excitatory and inhibitory currents, and to achieve the final firing rate close to the base firing rate $f = g(b_0)$. Connection strengths were thus converted to log-rate units using Eq.(26), where $w^{kk'}$ was finally multiplied by -1 to reflect inhibition.

To generate a sequence of spikes in this population, we simulated activity of the network forward in time computing currents injected into each cell from all previous spikes of all neurons. Each spike was assumed to inject the same PSP waveform, described by a temporal filter h_{PSP} [DIAGRAM] modeled as the difference of two exponentials with the rise time of $1ms$ and decay time of $10ms$ [**]

$$J^k_{t,inject} = \sum_{k'\neq k} w^{kk'} \sum_{\tau>0} h_{PSP}(\tau) n^{k'}_{t-\tau}. \tag{27}$$

(Given $1ms$ time step of our simulation, and the fact that a signal in a local cortical circuit of the size of $\sim 1mm$ would suffer $\leq 1ms$ time lag, we neglect delays in this simulation.) Additionally, each neuron exhibited refractory current with waveform h_{REFR} [DIAGRAM] modeled with exponential with decay time of $5ms$ [**]

$$J^k_{t,refr} = \Omega^k \sum_{\tau>0} h_{REFR}(\tau) n^k_{t-\tau}. \tag{28}$$

The network was then thus simulated forward in time with step of $\Delta t = 1ms$. All neurons were assumed to be electro-physiologically similar (i.e. have the same PSP profiles, Ω and base firing rate, etc.) Spikes were generated at each time according to Bernoulli distribution $n^k_t =$Bernoulli$[g(J^k_{t,spont} + J^k_{t,refr} + J^k_{t,inject})\Delta t]$.

Given a sequence of spikes, the fluorescence observations were generated using the setup in [VogPan]. Parameters for the model were chosen in accordance to our experience of analyzing a few actual cells [**]. Specifically, calcium decay time constant was $\approx 0.25s$, the ratio of per-spike calcium influx to stochastic fluctuations in calcium concentration was $\approx 3 : 1$, and the background of calcium concentration was chosen to be about 30% of per-spike calcium influx, thus corresponding to typical relatively low SNR of 1:3. Photon count per measurement was taken to be $\approx 3 \cdot 10^4$. The population of cells was generated with these parameters while assuming that all parameters may vary by at least 30% from cell to cell. Thus, we generated fluorescence for each cell using its unique cell of parameters, and produced sampled observations at frame-rate of 33Hz or 66Hz. [PARAMS-TABLE]

4.3 Including a sparse prior

4.4 Data

5 Other algorithms

XXX: this section is still under much development

5.1 Default algorithm: see [?] for details

Initialization :

1. Manually define an ROI around each neuron.
2. Average pixel intensity from all pixels in a frame to obtain $F_{i,t}$, i.e., the fluorescence intensity indexed by cell indentity i and frame number t.
3. Separately for each neuron, implement PFS as described in [?] (ie, with all cross-coupling terms turned off).

Recursion :

1. For each neuron, let the input, $\boldsymbol{u}_{i,t} = [\boldsymbol{s}_t, \{\widetilde{\boldsymbol{h}}_{i,t}\}]$, where $\{\widetilde{\boldsymbol{h}}_{i,t}\} = \{E[\boldsymbol{h}_{i,t}|\boldsymbol{F}]$ for all $i\}$.
2. Use PFS from [?] to learn $\boldsymbol{\omega}$

5.2 Population PF

1. same as 5.1
2. same as 5.1
3. particle filter step: this generalizes 5.1 by sampling not just on self-histories, but also cross-histories. because the transition distribution factorizes as above, we can sample from the optimal one observation ahead sampler, ie, $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}_t)$ (by approximating the likelihood term as a Gaussian in $[\text{Ca}^{2+}]_{i,t}$). in particular, to sample spikes after integrating out everything, one is left with:

$$n_{i,t} \sim p(n_{i,t}|\boldsymbol{h}_t)\mathcal{N}([\text{Ca}^{2+}]_{i,t}; \mu_{c_{i,t}}, \sigma^2_{c_{i,t}}) \tag{29}$$

where $\mathcal{N}([\text{Ca}^{2+}]_{i,t}; \mu_{c_{i,t}}, \sigma^2_{c_{i,t}}$ is a Gaussian approximation to the product of the likelihood and calcium update terms.

4. same as 5.1
5. estimate parameters using standard EM stuff as in [?]

it's not obvious whether this approach will outperform the approach outlined in 5.1

5.3 Gibbsish Population PF

this generalizes 5.2, by sampling conditioned not just on spike histories, but also on spike futures from other neurons. in particular, we want to sample spikes for one neuron conditioned on its own past spikes, and *all* spikes from all other neurons. having already completely an iteration, we now sample spikes according to:

$$n_{i,t} \sim p(n_{i,t}|\boldsymbol{h}_t, \boldsymbol{h}_{\setminus i,t+1})\mathcal{N}([\text{Ca}^{2+}]_{i,t}; \mu_{c_{i,t}}, \sigma^2_{c_{i,t}}) \tag{30}$$

where $\boldsymbol{h}_{\setminus i,t+1} = \{h_{1,t+1}, \ldots, h_{i-1,t+1}, h_{i+1,t+1}, \ldots, h_{N,t+1}\}$. we compute this using the method described in [?]. For future spikes, we let the posterior mean of the spike train from the previous EM iteration correspond to future spikes.

5.4 Gibbser Population PF

this generalizes 5.3, by sampling not just sampling based on previous EM iteration's mean posterior spike train, but actually doing a proper gibbs step within each forward pass. in particular, we do the sampling as described in 5.3 to initialize, and obtain a spike train for each neuron. we then iterate, sampling for each neuron, conditioned on the entire spike trains for other neurons. by gibbs, we will eventually converge to having sampled jointly from them all. i am not convinced that this is a particularly useful generalization of the 5.3.

5.5 Metropolis-Gibbs Population PF

this generalizes 5.4, by placing the Gibbs sampler just described into a MCMC approach. more specifically, we proceed as follows. for each neuron:

1. generate $M-1$ particles, sampling according to $p(n_{i,t}|\boldsymbol{h}_t, \boldsymbol{h}_{\setminus i,t+1}, F_{i,t})$ as described in 5.4.
2. add current path to population of particles and compute appropriately normalized transition probabilities
3. use standard forward-backward sampling algorithm for HMM's to sample $\boldsymbol{x}^*$ from this augmented space
4. compute $q(\boldsymbol{x}^*)$, the probability of sampling z using a forward-backward recursion
5. compute probability of acceptance, $r = \frac{q(\boldsymbol{x})p(\boldsymbol{x}^*)}{q(\boldsymbol{x}^*)p(\boldsymbol{x})}$, where $\boldsymbol{x}$ is the current path and $p(\boldsymbol{x}^*)$ is the posterior

we iterate this step until we accept a new spike train for a particular neuron, and then repeat for all subsequent neurons. alternately, we could randomly choose a neuron after each iteration. i have no idea which approach would be better.

speeding things up as this probably will take a while, we can do a number of things to speed it up

- skip generating particles step sometimes, ie, keep sampling from a particular HMM until prob of acceptance gets too small
- do an increment/decrement thing. in other words, after generating the first set of particles, with each additional acceptance, we simply eliminate a path (based on its likelihood). thus, we must only generate $M-1$ extra particles once.
- instead of using the standard hmm sampling to generate a new proposal, use Viterbi for PF (as described in [?], or the fast version of that approach (as described in [?] to generate a possible new sequence. then, if that one is not accepted, use standard hmm sampling to generate other proposals.

5.6 Ways we might make stuff better

these ideas potentially improve on all the above ideas.

1. Improved observation model — Identify ROI for each neuron using some ImageJ plug-in (or other such method). Instead of equally weighting all pixels, initialize pixel weights with first eigenvector computed using SVD, and then proceed as before. upon constructing SS, update spatial filter weights along with other parameters. An analysis of the kind of spatial filters should facilitate parameterizing the filter (potentially as a mixture of Gaussians). Alternately, an appropriate regularizing kernel may help. Note that if two or more neurons are in the same ROI, the observation model must be modified somewhat (but I'm not putting those details in here at this time)

2. Improved transition model — The most obvious improvement would be to allow additional time constants for the dynamics (either calcium or fluorescence). Perhaps the best way to decide what to include would be to take data for which we have both images and ephys, and fit a couple parametric models, and do cross-validation to compare which is best. The options would include: (i) an additional calcium state, (ii) temporal dynamics for the fluorescence, (iii) no noise on calcium, or (iv) some combination of the above.

3. Rao-blackwellized Particle Filter (RBPF) — This would amount to having a mixture of kalman filters (MKF), each conditioned on a particular spike train. As far as reducing unnecessary variance, I think it would be hard to justify not incorporating this into our model (see [**?**] for details). Note, however, that in the limit as $N \to \infty$, the particle approximation is perfect, whereas the RBPF (or MKF) is not, because we must estimate the likelihood as a Gaussian function of [Ca^{2+}].

5.7 these are not very good ideas, but i thought they were potentially worth writing down

1. Improved resampling — we could use a number of tricks here
   - sample $N_p' > N_p$ particles, but only resample using N_p particles, aka, prior boosting (see [**?**, **?**] for details).
   - Rejection sampling — We do not resample, but rather, only accept particles at each time. See [**?**] Section E for a summary (and refernces therein for details).
   - robust weighting (to handle outliers) (see review on particle filters)

2. MCMC with particle filtering — there are a number of ways to do this kind of thing
   - RESAMPLE-MOVE —
   - Metropolized Gibbs sampler — see [**?**] for details

3. data-augmentation for parameter estimation (ie, iterate estimating $p(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{x})$ to eventually converge to $p(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{y})$).

4. use algorithm in continuous-time particle-filter paper ([**?**]) for parameter estimation